REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 10 Jan 1997	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE Rule Based Systems	5. FUNDING NUMBERS N61339-96-0-D-0002
6. AUTHOR(S)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lockheed Martin Corporation Information Systems Company 12506 Lake Underhill Road Orlando, FL 32825	8. PERFORMING ORGANIZATION REPORT NUMBER ADST-II-CDRL-023R-9600238A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) NAWCTSD/STRICOM 12350 Research Parkway Orlando, FL 32826-3224	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT A - Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

This research activity examines the current state-of-the-art in Modeling the command decision process and implementing such models in software. The primary and initial target application is in automated command agents for DIS/ADS. This report was prepared for the Command Decision Modeling ADST II Delivery Order in accordance with the following documents:

a) Command Decision Modeling Overview (ADST-II-CDRL-023A-9600236)
b) Functional Description of a Command Agent (ADST-II-CDRL-023A-9600237)
c) Rule Based Systems (ADST-II-CDRL-023A-9600238)
d) Genetic Algorithms and Evolutionary Programming (ADST-II-CDRL-023A-9600239)
e) Petri Nets and Colored Petri Nets (ADST-II-CDRL-023A-9600240)
f) Neural Networks and Bounded Neural Networks (ADST-II-CDRL-023A-9600241)
g) Case-Based Reasoning (ADST-II-CDRL-023A-9600242)

14. SUBJECT TERMS Rule-Based Systems; Simulation; STRICOM; ADST-II DIS; KBS; AI;	15. NUMBER OF PAGES 18
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

DTIC QUALITY INSPECTED 1

ADST-II-CDRL-023R-9600238A
10 January 97

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

DELIVERY ORDER # 0018

CDRL AB03

RULE BASED SYSTEMS

ALL BUT WAR IS SIMULATION

FOR: NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32826-3224
N61339-96-D-0002
DI-MISC-80711

BY: Lockheed Martin Corporation
Martin Marietta Technologies, Inc.
Information Systems Company
12506 Lake Underhill Road
Orlando, FL 32825

DOCUMENT CONTROL INFORMATION

Revision	Revision History	Date
	Original release	7/15/96
A	Incorporated customer comments	1/10/97
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		

Table of Contents

PREFACE

This research activity examines the current state-of-the-art in Modelling the command decision process and implementing such models in software. The primary and initial target application is in automated command agents for DIS/ADS. This report was prepared for the Command Decision Modeling ADST II Delivery Order in accordance with the following documents:

a) Command Decision Modeling Overview (ADST-II-CDRL-023A-9600236)
b) Functional Description of a Command Agent (ADST-II-CDRL-023A-9600237)
c) Rule Based Systems (ADST-II-CDRL-023A-9600238)
d) Genetic Algorithms and Evolutionary Programming (ADST-II-CDRL-023A-9600239)
e) Petri Nets and Colored Petri Nets (ADST-II-CDRL-023A-9600240)
f) Neural Networks and Bounded Neural Networks (ADST-II-CDRL-023A-9600241)
g) Case-Based Reasoning (ADST-II-CDRL-023A-9600242)

1.0 OVERVIEW OF TECHNOLOGY AREA

Rule-Based Systems, originally known as Production Systems and later as a subclass of Expert Systems, are outgrowths of early problem solving research in Artificial Intelligence (AI). These systems are characterized by a general, underlying idea: problems in well-understood domains can be solved by structuring domain knowledge into IF-THEN rules by experts in the field. As the field of AI matured, techniques for encoding knowledge evolved into a second generation of expert systems which combines multiple knowledge representations and problem solving strategies within a single system. These systems are now known as Knowledge-Based Systems (KBSs). The underlying ideas for KBSs are: knowledge is central to problem solving and different models and problem-solving methods are needed for different aspects of the problem.

Encoding knowledge into rules is the most common form of knowledge representation for KBS [Gonzalez and Dankel, 1993]. One reason rule representation is popular is due to its simplicity: a rule is composed of an antecedent (IF condition) and a consequent (THEN clause). Rules can express a wide range of associations, such as situations and corresponding actions that are taken when these situations occur, and as premises and conclusions that result when those premises are true. Rules generalize relationships among objects of the domain. Facts about specific instances of objects are stored in a fact database and rules are evaluated against these facts. This representation of knowledge allows various types of knowledge to be encoded, for example:

1. facts: “A tank platoon is a platoon unit.”
2. information rules: “A platoon unit is commanded by a platoon leader.”
3. hard-and-fast or procedure rules: “Always stop the vehicle before dismounting DI units.”
4. problem situation rules: “If the unit arrives at the destination early, then halt until time to depart.”

A second reason for the popularity of the rule-based paradigm is that encoding domain knowledge into rules is similar to how human experts communicate their knowledge. This facilitates the acquisition of knowledge required to build a rule-base. Multiple experts can be interviewed and their expertise can be pooled and integrated into a repository of relevant knowledge. Analysis of legacy systems and written documents can also be used to acquire the necessary knowledge. In this way, a rule-base is built incrementally. However, the ease of knowledge acquisition is deceptive. Integrating the knowledge into a rule-base that can perform as well as a typical domain expert can be accomplished only with skillful knowledge engineering. That is, by focusing a domain so that the knowledge is specific, by writing and maintaining rules in tight chunks or sets of knowledge, and by editing and continually refining the knowledge base a successful KBS implementation can occur.

Essentially, a rule-base is invoked by presenting the KBS with a specific problem description or case, and by the KBS searching through its knowledge of rules and facts for an answer. The mechanism used to draw conclusions based on rules in the knowledge base and the data for the current case is the KBS's reasoning process or inference strategy. The inference strategy specifies the order in which the rules will be compared to the knowledge base and a way of resolving the conflicts that arise when several rules match at once.

The two strategies that control the sequence of rule firing are forward chaining and backward chaining. The forward chaining method of inferencing is a reasoning strategy that is also known as data driven, event driven, or bottom-up thinking. Starting from the known facts, rules are fired that derive new facts which in turn activate other rules thus forming an inference chain from the source state to the goal state. A rule is fired by matching the antecedent against the current fact-base to act out the consequent. The idea behind forward chaining is to find a sequence of rules that when given the data for a current case and known facts will lead to an answer. This approach is useful when initial facts are known about a problem situation and there are many possible goal conclusions. Problem domains, such as, design, configuration, planning, scheduling, and classifying are good candidates for the forward chaining strategy.

The other fundamental reasoning strategy is backward chaining. This strategy is also known as goal driven, expectation driven, or top-down thinking. It requires looking at the consequent or action parts of rules to find ones that would conclude the current goal, then looking at the left-hand sides or condition parts of those rules to find out what conditions would make them fire, then finding other rules whose action parts conclude these conditions, and so on. Basically, this method starts with the desired conclusion or answer to a problem statement and decides if the existing facts support the derivation of a value present in the knowledge base. Backward chaining is useful in domains where there are many facts that pertain to a problem situation but not all are necessary for deriving the solution. Domains such as diagnosis are well-suited for backward reasoning. Tactical decision-making also seems well suited to a backward reasoning paradigm since military behavior is often expressed in terms of goals to be achieved [Kwak, 1995]. It can also use forward reasoning when opportunistically using a given situation.

With either inferencing strategy, new facts are derived as solutions are sought. Once a new fact is derived, it remains true and becomes a member of the knowledge base. Adding facts permanently to the knowledge base is called monotonic reasoning. However, some systems allow new facts to be created by making assumptions based on currently known facts. These derived facts have an amount of uncertainty associated with them. This is typical of real-world problems where assumptions are made or defaults are taken based on the context or pragmatics of the current problem situation. New facts can arise that contradict previously asserted facts. The ability of the system to retract conditionally asserted facts is called nonmonotonic reasoning. Retracting of a fact may necessitate the removal of other, dependent, facts. Truth Maintenance Systems have been developed to maintain the integrity of the knowledge base when nonmonotonic reasoning is used. Problem domains, such as classification or diagnosis, can be implemented with monotonic reasoning. For more complex problems, and hence real-world problems such as command reasoning, nonmonotonic reasoning becomes a requirement.

Choosing forward or backward chaining is just one component of the inferencing strategy. The other component is deciding how rule selection conflicts will be resolved. Backward chaining systems generally use depth-first backtracking to select individual rules. Forward chaining systems generally employ sophisticated conflict resolution strategies for selecting among the applicable rules. Examples of conflict resolution strategies are: the first rule that matches, the highest priority rule, the most specific rule, the rule that has not fired before, an arbitrary rule, and firing rules in parallel. The most straight forward and easiest to implement is the first rule that matches strategy. Other strategies raise the complexity of rule creation and reduce the rule comprehension. The choice is not a research issue for command decision modeling but purely a development tradeoff.

The fundamental disadvantage of rule-based systems that affects command decision modeling is that as the knowledge base grows in size the likelihood that the system is still performing as an expert decreases. Complex, real-world domains like command decision modeling are knowledge intensive and require lots of detailed knowledge. Decisions are multiply constrained by knowledge from various sources, such as mission, terrain, enemy, troops, and time requirements. These sources are in themselves complex and difficult to model. Modularizing the knowledge base into rule sets increases the ease with which the rule-base may be modified and validated but restricts the complexity of the problem being modeled. The small rule sets can be thought of as experts that can handle detailed, specific problems but cannot efficiently solve problems together since they cannot communicate. This leads into the research area of autonomous agents: local experts that need to learn to work together.

Interest in rule-based technology has reached an equilibrium point for the most part as the paradigm moves from a research area to more practical implementation areas. Rule-based systems often serve as the foundation of more complex and/or hybrid reasoning schemes [see for example, Mall et al., 1995; Chaib-draa et al., 1993; Gonzalez and Ahlers, 1995]. This has lead to the development of knowledge based systems that are characterized by explicit modeling of different types of knowledge and by the use of a variety of problem solving methods that are geared toward particular subtasks.

2.0 NOTABLE IMPLEMENTATIONS AND VARIANTS

2.1 RPD Architecture

An early implementation of rule-base reasoning for command-level tactical decision-making is the RPD architecture [Chaib-draa et al.; 1993]. It is a hybrid architecture design to facilitate the coordination of intelligent agents to promote beneficial interactions and avoid harmful ones. This becomes an important issue since local decisions many times do have global impacts. The architecture is composed of three primary components: a reactive component, a planning component, and a deliberative component, hence the name RPD. The rule-based knowledge representation paradigm is used to encode knowledge for the reactive component that handles the reactive agent behaviors and the planning component that handles planning in the presence familiar situations. The deliberative component handles deliberative decision making for completely unfamiliar environments and does not employ the rule-based paradigm and thus will not be discussed further.

Under the RPD architecture, information from the environment is perceived by an intelligent agent. A planning action occurs if the information is structured as a goal, otherwise a reactive action occurs. Rule patterns specify reactive actions to the environment and the specification of goals. Whenever a situation cannot be handled by these rules, identification and recognition agents are needed. Furthermore, if the information is ambiguous or a goal needs elaboration due to an unfamiliar environment then deliberative decision making is performed. Once unambiguous, specific goals have been identified, planning can begin.

Planning is closely linked to perception, identification, and recognition. Perception either leads directly into planning or is first followed by identification or recognition before it leads to planning. Planning is followed by an action. The planning component always deals with familiar situations. The recognition process uses rules to recognize a certain goal or task. The rules for recognition adhere to the following format:

IF < signs> AND <situation> THEN <accomplish-goal>
or
IF < signs> AND <situation> THEN <execute-action>

Signs and the situation clauses in the IF-AND portion of the rule are the facts that define the current state of an agent's world. When signs become active for a given situation, a goal becomes active or an action is executed. The identification process uses Case-Based Reasoning which is discussed in the CBR section of this paper. Once goals have been identified, the identification process tries to determine the actions necessary to achieve those goals. The planner instantiates templates for plans to achieve those goals based from a set of plans issued from familiar situations. These templates may also be modified to fit the specific goals.

This form of rule execution is known as rule-based planning. It assumes that the state of the world at a given time is represented by a set of facts. The planning is characterized by a set of preconditions which are logical formulas expressed in terms of the facts that must be true before an action can be successfully executed. The effects are the set of formulas that will be true after the action has been successfully executed. The body is the set of subactions to be performed or subgoals to be achieved [Chaib-draa et al.; 1993].

To reach a specific goal state, the agent will execute rules that build a plan to achieve the goals. Case based planning (CBP) is used for unfamiliar situations. Two reasoning approaches are used within the rule-based system to construct a plan. Forward chaining is used to decompose actions into more primitive actions. Primitive actions are easier to use for creating different plans. Backward chaining is used to ensure that the preconditions of each action are satisfied. If any of the preconditions is not satisfied and not generated by some previous action in the plan, then another action is found that generates the precondition and includes it in the plan.

The RPD architecture seems well suited to a hybrid command reasoning architecture. However, no implementation details are given suggesting that the architecture has never been developed beyond the initial stage. It does serve as a possible blueprint for command reasoning systems of the future.

2.2 ITEMS

An early use of expert systems for Computer Generated Forces (CGF) is the Interactive Tactical Management System (ITEMS) [SikSik, 1993]. It provides an environment in which entities can interact with each other and the environment. The entities are modeled with respect to their underlying physical system and their intelligence. ITEMS supports, using rules, mission planning, opponent selection, and coordination within and between different echelons. For command and control, ITEMS supports company and battalion level commanders. The company level behavior is primarily characterized by platoons cooperating for a single purpose. The battalion level behavior is characterized by more complex coordination of forces with companies possibly cooperating on a task that does not directly benefit them.

Each echelon command level has an associated Command and Control Unit (CCU). The CCUs are templates that enable the assignment of tasks to each of the units and establish the chain of command. Tactical overlays can also be referenced by a CCU during scenario definition. This provides the unit with an initial position as well as a mapping appropriate to its command level. A set of doctrine is also associated with each CCU. The Company Coordination Doctrine provides the knowledge necessary to perform platoon coordination behavior. Typical coordination parameters include: messages and orders from the battalion commander, company survivability and kill indices, threat positions and status, company position, and mission information such as the current mission and the state of completion. Company behaviors encoded in rules are governed by doctrine according to the tactical situation [SikSik, 1993]. Example ITEMS company behaviors include: fire positioning, evasive maneuvering, withdrawing, halting, and finding of alternative routes.

The Battalion Coordination Doctrine provides the knowledge necessary to perform company coordination behavior. As an example of coordination behavior, a hasty attack behavior sequence is as follows [Siksik, 1993]:

- the battalion commander assigns, according to its doctrine rule base, a fire-base company and maneuver company
- directs the maneuver company to move to a fixed location relative to the enemy and suppressive fire
- directs the fire base company to outflank the maneuver company

Thus, planning is static in the sense that only predefined plans (via rules) are used and are not dynamically created from individual actions. The parameters considered are high level and include: company positions, company status and the battalion commander's orders to the company. The ITEMS battalion behaviors include: coordination of attack and maneuvers (e.g., hasty attack), artillery support, and resupply.

As mentioned previously, the knowledge base is written in the form of rules. The authors contend that as far as functional "how-to" knowledge is concerned, rules are very close to how experts organize their actions or ideas [SikSik, 1993]. The rule-base uses doctrine as its knowledge source and is organized into modules by echelon and internally by a tree structure hierarchy of rule sets. These rule sets partition the knowledge base into manageable units that describe a particular state or belief as appropriate to the behavior being represented. This limits the search space that a command agent has to use to decide upon an action.

The rules within ITEMS are based upon condition parameters calculated by the ITEMS control objects. Rule actions are divided into specific action categories, each of which is mutually exclusive. ITEMS uses forward chaining inferencing strategy since the state of the condition parameters is calculated and known every real-time iteration and since the number of possible responses is large. The conflict resolution strategy used is to pick the first rule that matches. This strategy is utilized because it is the most straight forward.

ITEMS has shown that rules can be used for CGF behavior. However, ITEMS exhibits the shortcomings of the expert system approach. High level behaviors are brittle, that is, when they are faced with a situation slightly different from the ones the rule-base was designed to handle, the rule-based approach fails. Since ITEMS cannot make decisions for situations it was not specifically design to meet, decisions within ITEMS are static and hard coded. This is not flexible enough for real-life combat situations. If enough doctrine is encoded into the rule-base, then ITEMS may be adequate for military training. Adding to the knowledge base leads to the another expert system shortcoming: a large knowledge base is difficult, if not impossible, to maintain and validate. This is why rules are often used in conjunction with other knowledge representation paradigms and problem solving approaches.

An important aspect of command decision modeling is planning. In ITEMS the mission planning aspect is alluded to but not really discussed, suggesting that this type of planning is not really addressed. A system that cannot handle novel problem situations and lacks planning is not robust enough to handle the command reasoning problem on its own. ITEMS may be modified to serve as a component of a hybrid system or as a foundation for a more complex expert system.

2.3 Concurrent Control and Coordination

Research has also been performed on investigating the use of concurrent control and coordination techniques for higher echelon commanders [Harmon et al., 1994]. These techniques are typically used for low level reactive behaviors. The goal of the research is to determine if concurrent control techniques can effectively be applied to upper echelon commanders.

A concurrent control system is composed of several individual control loops each working towards its own goal. As much intelligence as needed can be placed in these control machines. Rules are used to arbitrate and select the behavior to execute. Thus, various aspects of nonlinear control can be realized. The rule-based component becomes more valuable as more control loops are added to the system. This approach enables modularization of interacting control laws and limits the complexity to a tractable level.

The concurrent control methodology is based on the premise that multiple simultaneous goals must be evaluated in the execution of a mission [Harmon et al., 1994]. Here, each goal represents a single behavior that attempts to optimize its own performance independent of the other goals. An arbitration scheme resolves the competing or conflicting goals and either chooses a goal or combines the goals into a single goal to execute. Since the system is responding to all the active control loops, it is always responding to the entire pertinent situation and, thus, does not become trapped in an inappropriate control state or local minima [Harmon et al., 1994].

For command reasoning, a concurrent control commander evaluates the status of its task according to its orders and generates the necessary low level goals for its subordinates. Each individual task is mapped onto a concurrent control module. These tasks correspond to mission tasking, reporting, and command transfer. Individual control modules are also needed for each of these tasks to represent multiple concurrent missions and reporting tasks. In addition, separate arbiters are designed to interface to the different subordinates to ensure consistent commands are sent to all the lower echelons.

Concurrent control does show promise for dealing with the enormous number of possible combinations of situations that can occur during training exercises. The results are positive for limited scenarios such as movement to contact, attack by fire, and defend from ambush. Coordinated activity is demonstrated as well as transferal of command when a commander is disabled or killed. However, cognitive abilities such as situation assessment and planning were not addressed. For command reasoning, situational assessment and planning are requirements. The rules were used only to arbitrate the control and thus were not the focus of the research. In addition, the research effort primarily demonstrated the applicability of concurrent control to the reactive side of a command agent. Rules were also meaningfully applied to the plan monitoring and resource allocation problems associated with command entities.

2.4 Close Combat Tactical Trainer

The Close Combat Tactical Trainer (CCTT) simulates behavior from the lowest level vehicle behaviors through battalion. A rule based system was used for these behaviors in the CCTT SAF prototype [Ourston and Bimson, 1994; Bimson et al., 1994]. The decision to use a rule-based system for tactical decision making was based upon an analysis of the reasoning requirements (rules are suited to this kind of reasoning), extensibility, modification flexibility, and the need to isolate the reasoning component from the more algorithmic processes of platform behaviors and terrain analysis [Ourston and Bimson, 1994]. The SAF prototype focused on tactical decision making at the platoon level. While this is not useful for autonomous command agents, the approach can be scaled up to higher echelon behaviors. The prototype's purpose was to only illustrate the use of a knowledge base in making tactical decisions and arbitrating among competing alternatives of behavior from different sources.

It is a generally accepted conclusion that expert systems are too slow in reaching conclusions and, therefore, are not viable approaches for modeling real-time behavior. One reason for this conclusion is that real-time expert systems, such as RTworks, are agenda based systems. A system with an agenda is one where rules continue to fire until all potentially satisfiable rules have been tried and no new rules have been activated as a consequence of another rule firing. Rule processing time can vary from a few milliseconds to a few minutes depending upon the number of rules on the agenda. Another reason for slow execution speed is the complex pattern matching that expert systems such as RTworks perform. The SAF Behaviors prototype models entities and tasks as objects with slots for data values. Rules match on these slots as a precondition for rule firing. It is not known what rules will fire or in what order. With the addition of wildcards in the rule patterns, a great amount of rule processing can be performed, making the execution time difficult to predict. Algorithmic languages such as C or Ada could be used to execute the rules increasing the execution speed but at the cost of flexibility, maintenance, and increased development complexity.

The Behaviors prototype deals with mission, enemy, terrain, and troops over time (METT-T) concerns. Decisions concerning the who, when, where, why, and what of behavior are assigned to the rule-based behavior component while the how is contained in the algorithmic vehicle behaviors module. The execution problems of expert systems mentioned earlier is not as significant for tactical decision making since the response time requirements are less stringent, often in the range of seconds or even minutes. Tactical behaviors are characterized as more knowledge intensive but occurring less frequently than lower level platform behaviors such as moving a vehicle which is algorithmically modeled and real-time.

Most of the tactical decision making in the Behaviors prototype is done in response to situational interrupts. It is here that METT-T is addressed. The prototype only addressed Enemy and Time aspects of METT-T. Behaviors implemented include detection of enemy forces, direct small arms fire, direct antitank fire, indirect fire, fixed wing air attack, rotary wing air attack, defiles, and open terrain movement. The rules that respond to the situational interrupts are prioritized by threat and can be nested based upon this priority. The rulesets also detect termination conditions that allow an interrupted task to be continued. It should be noted that the prototype allows the human SAF operator to handle the more complex behavior via command overrides to the modeled entities.

The rules are organized into sets of modules for different echelons. Each module includes rules that represent and execute: orders, reports, command overrides, situational interrupts, and arbitration of behavior alternatives. Rules are further organized by category (movement, formations, reactive behaviors, defensive, offensive) and by Combat Instruction Set (CIS). Ultimately, rules were not used for the development system, not because they were not efficient enough or a close fit, but because knowledge engineering is still a research area and the project was under a tight development schedule. For behaviors guided strictly by doctrine, such as required to represent adequate training in CCTT, rule-based systems are appropriate.

In the CCTT SAF Prototype, rules were shown to be effective for platoon level tactical decision making, especially for multi-contingency situational awareness in which competing alternatives must be resolved. The rule structure closely resembles the kind of reasoning that is typical of tactical decision-making [Bimson et al., 1994]. The modularity of the rules was shown to be important for real-time execution. Most importantly, it was shown that the rules were not a bottleneck for system execution. The rate at which rule sets must execute and the rate at which situational interrupts occur during an exercise are relatively low and thus do not impact the overall execution time [Bimson et al., 1994].

One of the results of the SAF prototype effort was that it shed light on the representation and reasoning of METT-T. There is a difference between the behavior of command agents, typically judgmental METT-T, and event-driven METT-T [Bimson et al., 1994]. Judgmental METT-T data and behavior representations are more complex and performance intensive. Event-driven METT-T is the behavior typically characterized by the situational interrupt and thus was the concern of the prototype. Judgmental METT-T behaviors were not addressed by the rule-based system because of their complexity. Whether or not the rule-based paradigm was appropriate was not addressed.

2.5 Rational Behavior Model

The Rational Behavior Model (RBM) has recently been proposed as the representation for CGF behavior [Kwak, 1995]. RBM is a backward reasoning representation that can scale up to the higher echelon behaviors necessary for command agents. It is composed of three levels: strategic, tactical, and execution levels. Behavior control logic is located in the strategic level and controls the operation of the tactical level. The tactical level maintains the behavioral attributes for the system and represents the internal behaviors of the strategic level. The behavior interface is located in the execution level.

The strategic level is composed of backward chaining rules that form an AND/OR goal tree. The rules are written in Prolog without asserting any facts, only backtracking memory is used. The priority between valid goals is defined implicitly by the rule order. There is also a node priority for sibling sub-goals. Both priorities are statistically assigned a priori and together are used as the conflict resolution strategy when equally possible sub-goals are encountered. AND sub-goals have the same priorities but are of less significance since only OR sub-goals can potentially be conflicting. That is, more than one sub-goal might be eligible for a given condition. AND sub-goals are a sequence to achieve. If any AND sub-goals fail, the parent goal fails. Thus, multiple sub-goal activations cannot be allowed to occur.

The author contends that RBM is a natural representation scheme for military behaviors since it is well developed and organized. Military doctrine effectively expresses the global objective and sub-objectives and their associated actions and priorities. The author also states that the Army FM has effectively written the backward reasoning representation even though it gives the impression of a time-line based description [Kwak, 1995].

For command reasoning, the RBM provides the better domain match to target behavior description (FM or SME) [Kwak, 1995]. Application of the RBM to higher echelon behaviors is not mentioned in any detail. The goal-directed viewpoint is promising for higher echelon behaviors but the backward reasoning approach presented here seems to require all the knowledge a priori. There is no explicit support in the RBM for uncertainty, intentions, and partial fulfillment of goals that are necessary for autonomous command agents.

2.6 Context-Based Reasoning

A recent use of the rule-based paradigm for autonomous agents is using rules to represent contexts. The use of contexts overcomes some of the deficiencies associated with case-based reasoning, namely the requirement for a large case base containing specific cases. Context-Based Reasoning (CxBR) is an attempt to overcome these deficiencies [Gonzalez and Ahlers, 1995]. CxBR uses the concept of scripts to provide tactical knowledge to intelligent agents on the battlefield. These scripts provide the information necessary to perform situational awareness and the resulting actions (which may include switching to another context). The use of contexts is based upon the hypothesis that tactical experts use only the relevant information necessary for the task at hand, there are limited number of events that occur under a given situation, and that the presence of a new situation requires a change in the course of action [Gonzalez and Ahlers, 1995]. The key point in CxBR is that associating events and actions under a specific context eases the identification of such contexts and makes the behavior execution more efficient. In the domain of submarine warfare (where this work was developed) this is certainly the case. Only certain subsets of all possible situations are applicable under certain contexts. This work does not explicitly address the upper echelon behaviors that command reasoners need. However, the work's general architecture is applicable.

Because the contexts are general, the context base is usually much smaller than the case base. The script for a context supplies the steps required to carry out a specific action or recognize a new situation. CxBR can be used to define behavior at the mission level (mission contexts) and at the task level. Since the domain is narrow and specific, CxBR can be used to generate realistic behavior for training purposes. Contexts also include any special instructions and constraints that must be satisfied throughout the execution of the context.

The CxBR approach works under some of the following assumptions [Gonzalez and Ahlers, 1995]:

- Decision making is continuous and is heavily influenced by the continuous sequence of contexts. Each active context regulates the behavior of the agent and provides an expectation for the future. Contexts change as a result of external events and actions made by the agent. A context can be described as a well understood situation that has a predetermined set of procedures that must be carried out, either sequentially, methodically, or arbitrarily.

- Only one context is active at one time.

- Contexts are represented as intervals of time rather than time points. Contexts may be considered to be transitions to reach a goal or a goal themselves.

- Goals can be time points, but only to serve as transitions to other contexts.

- Only a limited number of things can take place in any single context.

- Certain cues exist which will indicate that a transition to another context is desirable.

- The presence of a new context will alter the present course of action and the applicable expectations to some degree. Using rules, the potential contexts and corresponding actions are associated with specific situations, simplifying the identification of a situation because only a subset of all possible situations is applicable. Similarly, the course of action to choose is easier under the recognized situation.

The CxBR approach is composed of a mission context, major-contexts, and sub-contexts. The mission context defines the objectives, constraints, and problems to avoid in the mission. It is usually defined in terms of lower level contexts and describes the political environment under which the mission is to be carried out. Major-contexts contain behavioral knowledge and context transition knowledge in the form of rules. Each major context is an individual task that is necessary to meet the overall goals of the mission. A context is activated by retracting from the fact base the identifier of the current context and asserting the identifier of the new context. The initialization procedure of the new context is also called which in turn activates new context monitoring rules. Any agent attributes (parameters) may also be modified if necessary. The context is set up as a class that can used for pattern matching in a rule-based system. The context contains initializer functions, an objective slot, a compatible-next-major-context slot, and a compatible-sub-context slot. The objective slot is stated in general terms and references the frame that has attributes that are the goal of this context. The Compatible-next-major-context attribute contains the allowable context transitions. The Compatible-sub-context lists all the sub-contexts that are compatible with the current context.

As mentioned previously, the recognition of the situation is done through pattern matching rules. The use of active context pattern in the premise modularizes the active rules to make the inferencing more efficient by reducing the possible action space. Monitoring rules handle the transition from one context to another. They fire continuously as long as the context is active. They monitor parameters which are relevant to the continued execution of the current context. When the actions of a monitoring rule are executed, the current context identifier is changed and the new context identifier is asserted along with any initialization.

CxBR is an effective and efficient mechanism for imparting sufficient intelligence to intelligent agents to achieve training objectives [Gonzalez and Ahlers, 1995]. Sufficiency is defined by the ability to behave as a human enemy would in a very specific and narrow domain. CxBR can be used to accurately represent tactical behavior from a qualitative standpoint [Gonzalez and Ahlers, 1995]. An important conclusion is that through the prototype, CxBR was shown to be compatible with a distributed simulation environment. This is an important consideration for military command reasoning.

CxBR is done in the rule-based paradigm but is not a pure rule-based mechanism since it contains objects and context identifiers. Through experimentation it was shown to be slightly faster than a pure rule-based implementation. As the size and scope of the domain expand this will have more of an effect. For a limited prototype, the pure rule-based approach is more concise because of the CxBR overhead of the context objects. As the situation becomes more complex however, the overhead will become a smaller part of the knowledge base. For the rule-based approach the rules required become more numerous, complex, and inter-related.

CxBR has the advantage of encapsulating all the facets of tactical knowledge for a small slice of the domain. By modularizing the knowledge in this way a more efficient execution can be achieved. Also limiting the number of possible transitions between contexts is important for execution

The prototypes used in this research are primarily reactionary in nature and thus the planning capabilities were rather limited. Nevertheless, from a conceptual standpoint, planning is quite consistent with the general CxBR approach. Thus, the CxBR approach shows promise for scaling up to higher echelon behaviors.

3.0 APPLICABILITY TO COMMAND DECISION MODELING

Depending upon training objectives, a rule-based paradigm may not be the best choice for command reasoning due its brittleness in domain coverage. Situations in the domain not covered by the rules cause the command reasoning to fail. This is definitely a limitation of the paradigm since tactical decision-making involves dealing with an infinite combination of situations, such as number, types, and positions of entities, weather, and terrain. A more promising approach that has recently shown promise is the use of *intelligent agents* in support of command decision making. These consist of relatively small rule-bases that are used to solve well-defined and constrained subproblem areas. These intelligent agents contribute to the problem solving process as the conditions that apply to there domain expertise are met. In this way, the problem solving process departs from the monolithic architecture of early systems and incrementally makes use of domain knowledge as it applies to the problem at hand.

Rule-based systems are not good candidates for embedding into general real-time applications. Their response time is slow relative to the needs of the real-time application. However, command reasoning models high level command decision making which is slow relative to the overall execution time of the simulation [Bimson et al., 1994]. Therefore, rule-based systems are a possibility for this domain.

Rules are a natural method for representation of doctrine which is expressed in terms of conditions and sequences of events to perform. In addition, the ability to modularize rules makes them a good choice for expressing tactical knowledge. Rule-based reasoning is adequate for command reasoning in familiar situations and for representing the reactive behaviors component of a command agent. However, when situational awareness and planning are required in addition to doctrine knowledge, the rule-based paradigm alone is not sufficient. For real-life command reasoning a hybrid architecture using other reasoning mechanisms, such as case-based reasoning, is necessary.

4.0 CONCLUSIONS

[Optional. Not Required]

5.0 REFERENCES

Bimson, Kent; **Marsden**, Craig; **McKenzie**, Frederic; **Paz**, Noemi, "Knowledge-Based Tactical Decision Making in the CCTT SAF Prototype", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 293-303.

Chaib-draa, B.; **Paquet**, E.; **Lamontagne**, L., "Integrating Reaction, Planning, and Deliberation in Architecture for Multiagent Environment", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 45-55.

Gonzalez, Avelino J.; **Dankel**, Douglas D., *The Engineering of Knowledge-Based Systems Theory and Practice*, Prentice Hall, Englewood Cliffs, NJ, 1993.

Gonzalez, Avelino J.; **Ahlers**, Robert, "Context-based Representation of Intelligent Behavior in Simulated Opponents", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 53-61.

Harmon, S. Y.; **Yang**, S. C.; **Tseng**, D. Y., "Command and Control Simulation for Computer Generated Forces", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 263-271.

Kwak, Se-Hung, "A Comparison Study of Behavioral Representation Alternatives", *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 529-539.

Ourston, Dirk; **Bimson**, Kent, "Integrating Heterogeneous Knowledge Processes in the SAF Behaviors Implementation", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 333-341.

SikSik, D. N., "Intelligent Computer Generated Forces Through Expert Systems", *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation,* Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 3-9.